

# NAVY AVIATION MAINTENANCE INTELLIGENT FREE-PLAY TROUBLESHOOTING SIMULATIONS: CAPTURING THE AVIATION MAINTENANCE DECISION PROCESS

*Terrell N. Chandler Ph.D., Galaxy Scientific Corporation, Atlanta, GA*

## Introduction

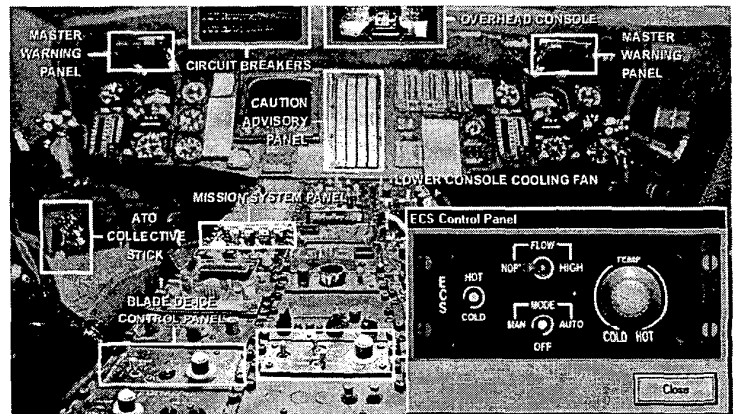
Over the last two years, Galaxy Scientific has designed and implemented intelligent free-play troubleshooting tutors for naval aviation maintenance personnel. The main objective of these tutors is to strengthen troubleshooting skills while completing maintenance of H60 helicopter subsystems. Troubleshooting skills involve unique decision making processes that demand inquiry techniques coupled with well-founded understanding of the target system. An aviation maintenance technician must possess the skills to recognize what a symptom reveals about the system and, from this collection of indicators, decide which next inspection or test will best reveal the faulty component causing such symptoms.

Galaxy employs a proprietary tool, Tutorware™, to efficiently develop intelligent free-play tutors. Tutorware™ originally had been conceived as an aid to Subject Matter Experts (SMEs) in their attempts to develop intelligent tutors. It became apparent that providing a development tool to SMEs was not sufficient to build a successful tutor. There are software and training considerations beyond subject matter expertise one must understand in order to complete a successful instructional simulation. As a result, Galaxy has repurposed Tutorware™ to aid educational technology software engineers, while working in concert with SMEs to effectively build these tutors.

This paper is divided into three parts. The first part is a description of the decisions a technician needs to make in order to become an expert maintenance troubleshooter. The second part will discuss some of the pedagogical and technical issues a developer must address when working with SMEs. The third part of this paper will discuss future trends in the instructional

simulation field. Two areas will be covered. The first will be the trend to use intelligent simulation-based tutors to train complex decision-making skills. The second will be the practical challenge of migrating these sophisticated media-rich tutors to the Web. Both technical and pedagogical issues will be covered as it applies to the unique domain of aviation maintenance troubleshooting.

## Background



**Figure 1: The panels highlighted in yellow can be accessed to troubleshoot an ECS system on the H60 Helicopter.**

Intelligent Free-play Troubleshooting Tutors (IFT Tutors) are high fidelity simulations of major subsystems coupled with intelligent coaching of student activities. Examples of large complex systems developed within Tutorware™ include helicopters, nuclear power plants, and automobiles. The development of 13 intelligent free-play tutors for the US Navy H60 helicopter is the most extensive application of Tutorware™ to date.

Each IFT Tutor includes an information area, an introductory tutorial, and a free-play troubleshooting area. The information area is where system components and location information are graphically displayed and described. The tutorial features the Microsoft Agent "Troubleshooting Tom" who walks the student through a typical problem and demonstrates the tutor features. The free-play troubleshooting area has a series of problems where each student is challenged to determine the faulty component.

Within each simulation, students can practice and compare their troubleshooting skills against an expert. Students can move freely about the system performing procedures, tests, and visual checks to first verify the identified symptom and then isolate the actual faulty component. They can flip switches, read gauges, and manipulate CRT screens. As an example, in the H60 Environmental Control System (ECS), students can manipulate the panels related to automatic and manual modes of the ECS (See Figure 1) to determine where the fault may be.

Students receive feedback if they are about to perform a hazardous act or have deviated too far from the task. Instructional aids include the fault symptom description, procedural advice, student action history, and pilot query. At the close of a problem, students can review their actions against the action an expert would have taken, compare their time against an expert's time to diagnose the fault, and review different kinds of errors (i.e., redundant actions, procedural errors, and hazardous errors).

Tutorware™ is composed of a system model, a student model, an instructor model, and an expert model. The system model drives the simulation. It is a functional flow diagram that represents the interrelationship between components in the system. Maintaining the simulation and triggering coaching feedback to the learner is generative. As the learner freely manipulates the simulation, activities affecting one part of a system will propagate through the functional network to affect other parts of the system, triggering coaching advice at the appropriate time.

The student model monitors what actions a student takes during an individual practice problem. It also maintains how well the student is progressing through a course of practice problems. The expert model embodies expertise needed to troubleshoot the system. This knowledge is physically captured within the functional flow diagram, troubleshooting procedures, troubleshooting techniques and fault knowledge such as system faults, procedural errors, and hazardous errors.

Troubleshooting techniques are derived from the split half technique [1]. This technique is used to determine what is the next best test to do to find the faulty component. These recommendations are made based on efficiency parameters, such as cost and time, to perform the test, as well as the most likely faulty component.

The instructor model compares the expert knowledge with student actions and provides coaching where appropriate. Functional advice, procedural advice, general instruction for interacting with the tutor, and warnings about hazards actions, procedural errors or new faults arising in the system are some of the types of coaching the instructor can provide.

Two types of decision making are involved in IFT tutors: the decision making that one is trying to teach the student and the decision making involved in building the tutors. The latter involves extracting the appropriate knowledge from the SME so that the developer can represent the simulation and the coaching to achieve an environment where the former happens. This second form of decision making is very much a collaborative and communication process between the SME and the developer. We will discuss each of these forms of decision-making activities in turn.

## **Decision Making Involved in Solving a Troubleshooting Tutor**

Typically, a troubleshooting session is divided into two parts: verification of the fault and identification of the faulty component. The flight crew reports a "gripe", or symptom they experienced during a flight exercise, to the

maintenance technician. The maintenance technician is responsible for checking the reported symptom to validate that the anomaly indeed occurred. It is often the case that reported errors cannot be repeated by the maintenance crew. The incident is noted and the condition monitored until the anomaly reoccurs.

During the process of verifying the fault, the maintenance crew runs a series of checks and inspections that tell the maintenance personnel that a) the fault is valid and b) the likely subsystem (e.g. electrical, mechanical, hydraulic, sensor) causing the problem.

Once the scope of the search has been narrowed to a specific subsystem, the maintenance technician will run more formal tests and inspections to determine the exact component that needs to be replaced.

The key to successful troubleshooting is to know the workings of a system well enough to be able to recognize anomalies in the system and know what those anomalies mean in terms of what might be causing those conditions. At each step of the process, the technician will be presented with new information that either validates his or her theory for what is wrong or discounts it. Given all of the information that the technician has gathered about the system to date, the technician must decide what is the next best test, check, or inspection to do to further narrow the search. Sometimes validating that one is on the right track includes conducting a test to verify that a certain subsystem is definitely not the problem. For example, a technician may run a test to verify that the sensors are definitely not the problem and he or she should be looking instead at the electrical subsystem.

Skill areas that technicians need in order to be successful troubleshooters include:

- Knowing the component function, location, and how the components interact
- Recognizing symptoms
- Identifying failures
- Usage of test equipment
- Reading test results
- Analyzing test results

- Using and understanding technical documentation
- Reading and following directions

Other factors that are important when judging whether a technician has performed well are time, safety, and accuracy. For example, maintenance tasks have to be performed with expedience since aircraft cannot be littering a deck of an aircraft carrier when planes are trying to land. In addition, on deck aircraft maintenance can be a hazardous undertaking. These two factors need to be in the forefront of a technician's mind while performing troubleshooting tasks. This means the technicians cannot be in such a hurry that they electrocute themselves while trying to set up their test box, but their troubleshooting techniques need to be focused, so that no extra steps are taken to locate the problem. This is where good decision making comes in, given the information at hand, the technician knows what is the next best test to perform.

## **Decision Making Involved in Building a Tutor**

The most difficult challenges to the CBT industry include:

- Extracting knowledge from an expert
- Configuring the raw content so that it is understandable to the learner
- Designing the computer program so that it comprehends the learner's current state of understanding
- Designing the computer program so that it responds appropriately to the learner's current state of understanding

This section will focus on the first two bullets: knowledge acquisition and translating that knowledge into a working IFT Tutor. The latter two bullets will be addressed in the section, "Expanding the Horizons of Intelligent Free-play Tutors".

There have been and there continues to be large efforts to automate the process of knowledge

acquisition. A recent effort is the ENGRAMS<sup>1</sup> project sponsored by the Air Force Research Lab at Brooks Air Force Base [2]. Here they are translating Air force doctrine into content objects that will be used as Curriculum Elements (CEs) within training systems.

For most CBT, including the IFT tutors, knowledge acquisition is a human-to-human process. The IFT development staff, in coordination with SMEs, has created an elaborate set of documents that the SMEs fill out to specify

- Problems to be solved
- Aspects of the simulation to build
- And expert advice to give to the student

Even with a cadre of tools and documents there are many difficult areas to overcome before an IFT tutor is successfully built. These are outlined below:

Selecting the right system for training troubleshooting. Certain systems and problems for those systems are better suited for troubleshooting practice than others. Some systems may be too simple or too procedurally lock step to afford the reasoning skills needed in decision making.

Selecting the right set of problems within a system. For example, long procedures with many redundant actions are generally not optimal if the goal is to teach reasoning skills. It is better to have problems where reasoning about the task is frequent and redundant actions are minimized. If the procedure is too long, one needs to decide if any parts of the process can be stubbed with out loosing the essence of what is being taught. Another example is the SME recognizing what is hard to simulate and what is easy to simulate. A popped circuit breaker may be easy for a technician to troubleshoot, but it may not be easy for a developer to simulate. It is the responsibility of the developer to help the SME choose problems that meet the learning objectives of the training while remaining within the scope of the project.

Selecting the right scope for problem solving. When trying to simulate a large complex system on the computer, the level of granularity is very

important. One must ask what the right level of fidelity should be to give the feel for the work the student must learn to do. If the fidelity is too high then a) the focus tends to be on the details of the activity rather than on the reasoning involved in the activity and b) the project is at risk of becoming too cumbersome and too costly. If the fidelity is not high enough, then the skills are not realized and transferred. One needs to wear a fish-eye lens when making decisions about simulation scope. One needs to be able to focus on the important aspects of the simulation and problem while blurring less important troubleshooting details.

Predicting how the system will react if a student deviates from the prescribed procedural path. What should be in focus is especially important when trying to decide how much free play one should allow the student. It is virtually impossible for the SMEs to know what would happen for every possible deviation a student might take when manipulating a system. A SMEs reaction to a developer asking, "What if the student does this ..." would often be "I don't know, no one in their right mind would ever do that." How much does one let the student explore and how much they are constrained to the task is a question revisited throughout the process. Since every deviation means implementing a correct simulation of the results, practicality often wins. For example, hazardous actions and certain procedural error are flagged and prevented from occurring. Still, the student is given a great deal of latitude when playing with the tutor.

Being consistent across procedures and within a procedure. Inconsistencies tend to arise when SMEs are writing many problems with similar procedures over an extended period of time. In addition, translating a troubleshooting procedure so that the steps have a one-to-one correspondence with the simulation steps is not as straight forward as one would think.

Some of the consistency issues can be minimized through tools where information entered earlier in the process can be replicated at later points. These techniques are limited however, because simulation and problem development for each tutor is a creative process. Each new tutor

---

<sup>1</sup> ENGRAMS = Empirical Normative Grammar for Representing Acquired Memory and Skill

presents new challenges that the SME and developer have to tackle together.

## In Summary

The key to successful troubleshooting is to know the workings of a system well enough to be able to recognize anomalies in the system and know what those anomalies mean in terms of what might be causing those conditions. The challenge is to build IFT tutors so that this type of context-specific recognition and understanding happens. To achieve this, one must choose the right systems to model, the right problems to solve, and the right level of scope to represent. Too much detail makes the problem-solving exercise cumbersome. Too little detail fails to mimic the actual circumstances within which the problem-solving task occurs. Finding the right balance is very much a team effort.

## Future Trends

### *Expanding the Horizons of Intelligent Free-play Tutors*

The next generation Tutorware™ will expand the current capabilities from a functionally-based intelligent tutoring system for maintenance troubleshooting to a series of decision-based tutors that can be applied to a variety of decision-aiding scenarios.

The reasoning structure of Tutorware™ is primarily based on representing the system and inferring the student's thinking through manipulation of that system. Sometimes a system does not lend itself to the best reasoning structure. For example, in the H60 Stabilator, all the subsystems (electronic, sensor, hydraulic, mechanical) went through the amplifier. As a result, the coach would advise the student to check the amplifier first. All things being equal, this is not bad advice since the amplifier is often the faulty component. However, in actual practice, Navy personnel initially conduct certain checks to determine which subsystem they should focus their troubleshooting upon. The Stabilator system model could represent the four subsystems to facilitate accurate coaching, but this representation

tends to get large, complex, and cumbersome to manipulate.

Modeling the system is best for homogeneous systems such as an electrical system or a hydraulic system. For heterogeneous systems, as represented by the Stabilator, a better approach is to overlay a procedurally derived reasoning structure.

The next generation Tutorware™, due out in December, 2000, will do just that - overlay a flexible procedurally derived reasoning structure over the functionally-based simulation. The functionally based simulation will be generalized to accommodate a wider variety of complex scenarios - scenarios such as crisis action planning related to weather or terrorism, and decision making involved in Epidemiology.

### *Web-based IFT Tutors and ADL SCORM*

There is a mandate from high-level officials within the Department of Defense (DoD) to migrate all training materials to Web-based delivery.

In anticipation of the DoD mandate, Galaxy has successfully migrated IFT Tutors to be launched from a server over the Internet and manipulated by a student through a Web browser. The next generation of IFT Tutors will all be Web compatible.

The DoD has established the Advanced Distributed Learning (ADL) Initiative to develop a DoD-wide strategy for using learning and information technologies to modernize educating and training. The ADL initiative has defined high-level requirements for learning content. These requirements, outlined below, are documented in the Sharable Courseware Object Reference Model (SCORM), Version 1.0 [3].

- **Accessibility:** the ability to access instructional components developed in one location and deliver them to many other locations
- **Interoperability:** the ability to use instructional components developed in one location with one set of tools or platform in another location with a different set of tools or platform (Note:

there are multiple levels of interoperability)

- **Durability:** instructional components that do not require redesign or recoding to operate when based technology changes
- **Reusability:** the design of instructional components so that they can be incorporated into multiple applications

These requirements are embodied within the specifications for the ADL's Learning Management System (LMS). Content providers are expected to interface with the LMS successfully. The SCORM requirements are currently in the testing phase. Galaxy has been and will continue to be heavily involved in the ADL SCORM initiative at all levels. Recently, The Orlando ADL Co-Lab has awarded the Naval Post Graduate School and Galaxy funds to use one of Galaxy's web-based products, the Safe Maintenance in Aviation Resource and Training Center (SMART Center), as a test case for SCORM compliance.

For courseware to successfully operate on an ADL-compliant Learning Management System (LMS), the following five areas must define the courseware:

- Course Structure Format XML Document
- Sharable Courseware Objects (AUs)
- Metadata (also written in XML)
- Data Model
- Runtime environment (API)

For each course, a Course Structure Format (CSF) document must be written. Course Structure Format is an XML document that defines the structure of the course so that it can be moved from one LMS system to the next. The CSF also defines the course's intended behavior so that, for each individual student, the LMS can control the student's progress through the course.

The CSF describes a course by describing the interrelationship between Assignable Units (AUs). AUs serve as the smallest component of a course which can be packaged with sufficient information to be reusable and accessible by the course itself,

as well as other courses [3]. Assignable units make up the shareable courseware objects of the SCORM model.

Each AU is described by metadata. Metadata is used to describe key features of the AU so that the CSF can reference these objects in its course description. The metadata is also used within the courseware repository as a means for cataloging and searching these objects for reuse in other courses.

The data model is the definition of the data exchanged between the LMS and the client computer through which information about student performance is passed.

Run-time environment is how the LMS passes content from the server to the client. The Application Program Interface (API) provided by the LMS is used by the content to communicate with the LMS.

There are many issues that arise when attempting to make a complex training application, such as an IFT Tutor, SCORM compliant. For example, one of the main challenges of the Web-delivered IFT Tutor is choosing the right level for the Sharable Courseware Object. Since timing is an important component of effective coaching and simulation, instructors do not want download time interfering with the timing of the simulation (e.g. a simulated light turning on, a CRT screen updating, etc.). Oftentimes, the first indication of a faulty component is when something does not turn on or off or move up or down within the appropriate timeframe. Assignable Units (AUs), therefore, must be selected so there is no confusion between the *physical updating* of the tutor software and the *simulated updating* of the system.

Another example is the issue of reusability. At one level, it makes sense to treat individual problems in an IFT Tutor as independent units that can be mixed and matched with other courseware units. At another level, it may also makes sense to make available specific media, like video, that an instructor can lift from one resource and re-purpose to support other instructional needs. However, as one opens the content to smaller and smaller sharable units, more and more issues arise. A simple example is writing and presentation style when moving specific content sections from one

course to another. Mixing and matching of different interfaces is also a consideration. As instructors mix and match different lessons from different vendors, students may have to learn a new interface for each new lesson presented.

Reusable media also has its own set of interesting considerations. In what format should a graphic be saved or stored, for example? If it is saved as a static picture rather than a collection of objects, then resizing the graphic becomes a problem. Storing and describing the same picture half a dozen times, with the only variant being its size, is a condition likely to arise.

Early in the H60 program, Galaxy had an opportunity to deliver an ITF Tutor on the Stabilator to four of the services. While the problems themselves could be ported over with few changes, all media needed to be swapped out to reflect the uniqueness of each service. There were also some differences associated with equipment location and equipment usage. For example, each service had a different set of test procedures for the SSLTS test set. All of the problems had to be modified to reflect the differences of the test set for each service.

If we use this as a point of analysis for shared courseware, we find that AUs and metadata really represent two different bodies of descriptive information. The runtime AU informs the LMS what to deliver and when to deliver it. A runtime AU may be defined at the level of the problem.

The course sharing of metadata is the basis for converting existing instruction into instruction for new purposes. Metadata AUs could be at the media level.

The ADL-SCORM initiative is a very ambitious undertaking. The ADL community is asking all stakeholders of government contracted CBT to adopt a new paradigm in the way courseware will be developed and delivered. The above discussion highlights but a few of the issues training community will need to resolve. The challenge is even more acute when considering conversion to SCORM compliance of legacy courseware.

## In Summary

Future trends dictate that training of all types will be web based. This has rewards because the training will be much more accessible to a larger group of students. It will be particularly beneficial On-the-Job training where travel and time off task are costly. There are many, many unresolved issues, especially when considering initiatives such as ADL SCORM, that must be tackled in order to provide quality training over the internet.

Other trends are the use of artificial intelligence techniques to address the difficult challenges of knowledge acquisition, knowledge representation, recognition of the student's state of understanding and advising the student appropriately. Expanding the capability of Tutorware™ to support a wider variety of simulated scenarios beyond maintenance troubleshooting is a realizable near term goal for Galaxy.

## References

1. Goldbeck, R. A., B. B. Bernstein, W. A. Hillix, M. H. Marx, 1957. "Application of the half-split technique to problem-solving tasks". *Journal of Experimental Psychology*. Vol. 53, No. 5.
2. Regian, J. W., 2000. Knowledge Representation Technologies for Human Performance. Slide Presentation. Air Force Research Lab, Brooks Air Force Base, San Antonio, Texas.
3. Dodds, P. (Ed), January 2000. *Sharable Courseware Object Reference Model*. Version 1.0. Advanced Distributed Learning, Alexandria, VA.